Prikupljanje i prikaz podataka o izvršavanju programa

1. Uvod
   * + kratak opis o čemu će biti reči u daljem tekstu
     + iako vidim da je popularno po master radovim ada se piše po poglavljima ovde (tipa, u poglavlju X je opisano to i to), ja bih uvod radije sročila kao pričicu koja prati rad
     + ovde bih dodala na samom početku i na samom kraju značaj teme kao takve i naravno značaj mog doprinosa ( na kraju zbog efekta)
2. Profajliranje i pokrivenost koda – teorijski uvod u osnovne pojmove i kako sad radi GCC
   1. Profajliranje i pokrivenost koda
      * Ovo je čisto teorijski deo - uvod u tematiku i pojmovi bez kojih se ne može dalje
      * Manje više bi ostao kao u radu za Etran, možda malo proširen i preformulisan
      * Počeo bi podelom na statičku i dinamičku analizu, i naravno na kraju bi se priča svela na objašnjenje najužih pojmovaČ profajliranje, instrumentalizacija i pokrivenost koda.
      * Zbog promene naslova, koja nažalost dosta utiče na suštinu, ja bih ovde dodala kratki hint za sledeća dva poglavlja, ali bez mnogo objašnjenja (npr. Podaci se mogu prikupiti na kraju ili tokom rada. E, ovo drugo je ono što nema, pa ja dodajem)
   2. Postojeća rešenja u okviru GCCa
      * Opis kako radi GCC instrumentalizacija, libcoverage i GCOV.
      * Ovde je dobar trenutak da se pomenu mane postojeceg statickog pristupa
3. Zacetak ideje i trnoviti putevi
   1. Ideja – dinamicki pristup!
      * Uvod u moj projekat
      * Šta je ovde drukčije i bolje
      * Samo teorija, bez detalja kako tačno radi šta
   2. Razmatrana rešenja
      * Dva puta koja su se predamnom bejaše otvorila – da li napadati GCOV alat ili menjati biblioteku
      * Kako i zašto sam odabrala ovo što sam odabrala
      * Lepa pričica da se pokaže da se ipak ulagalo malo mozga u projekat
4. Implementacija i analiza ucinjenog
   1. Implementacija
      * Biblioteka
      * GUI (signali za prikupljanje podataka, generisanje izvestaja)
   2. Demonstracija u uputstvo za upotrebu
      * primer rada biblioteke I GUI-ja sa slikama
      * dobar moment da se naglasi da rad ima primenu na bilo koji kod
      * ne znam jel smem pominjati digitalnu i ko ga sad koristi
   3. Performanse
      * da li smo postigli cilj
      * da li možemo isto što i pre, pa i više
      * memorija I bezbednost – test sa Valgrindom
      * složenost – vremenska i prostorna
      * jednostavnost upoterebe
      * ne bi bilo loše ovde pomenuti LLVM i njihovu runtime instrumentalizaciju (LLVM jede memoriju ko lud, jer podatke dampuje u fajlove ali i ugrađuje u samu binariju koja zato naraste mnogo i pravi probleme, dok GCC sve dampuje u fajlove pa je binarija mala).
   4. Primena
      * Gde bi sve ovo moglo da radi
      * Ne znam koliko smem odavati na čemu je testirano I na čemu radi
      * Ideja: Ako bi se ovakav jedan alat unapredio I ugradio npr u pejsmejker da signalizira da nešto ne radi kako treba, to što je runtime prikupljanje moglo bi nekome spasiti život
5. Zaključak
   * + Šta je urađeno
     + Koji je značaj toga što je urađeno (gde sad radi – onliko koliko smem da kazem)
     + Šta bi još moglo da se uradi:
       - Ideja: Ako bi se ovakav jedan alat unapredio I ugradio npr u pejsmejker da signalizira da nešto ne radi kako treba, to što je runtime prikupljanje moglo bi nekome spasiti život
       - Moze mala komparacija sa LLVMom – tipa da se analizira sta je dobro i da se malo unapredi po ugledu na LLVM
6. Literatura
   * + Imam za sad jedan dobar rad (onaj iz molbe)
     + Sajtovi na GCC, GCOV I tako to
     + Ideja: možda master nekog kolege iz firme koji je radio ovo pre mene (npr. Nikola Prica, Darko Ristivojević, Goran Vukov)